Interactive Visualization of High-Dimensional Petascale Ocean Data

David A. Ellsworth*
CSRA at NASA Ames Research Center

Christopher E. Henze†
NASA Ames Research Center

Bron C. Nelson ‡
CSRA at NASA Ames Research Center

ABSTRACT

We describe an application for interactive visualization of 5 petabytes of time-varying multivariate data from a high-resolution global ocean circulation model. The input data are 10311 hourly (ocean time) time steps of various 2D and 3D fields from a 22-billion point 1/48-degree "lat-lon cap" configuration of the MIT General Circulation Model (MITgcm). We map the global horizontal model domain onto our 128-screen (8x16) tiled display wall to produce a canonical tiling with approximately one MITgcm grid point per display pixel, and using this tiling we encode the entire time series for multiple native and computed scalar quantities at a collection of ocean depths. We reduce disk bandwidth requirements by converting the model's floating point data to 16-bit fixed point values, and compressing those values with a lossless video encoder, which together allow synchronized playback at 24 time steps per second across all 128 displays. The application allows dynamic assignment of any two encoded tiles to any display, and has multiple interfaces for quickly specifying various orderly arrangements of tiles. All subsequent rendering is done on the fly, with run time control of colormaps, transfer functions, histogram equalization, and labeling. The two data streams on each screen can be rendered independently and combined in various ways, including blending, differencing, horizontal/vertical wipes, and checkerboarding. The two data streams on any screen can optionally be displayed as a scatterplot in their joint attribute space. All scatterplots and map-view plots from the same x/y location and depth are linked so they all show the current brushable selection. Ocean scientists have used the system, and have found previously unidentified features in the data.

Index Terms: J.2 [Physical Sciences and Engineering]: Earth and atmospheric sciences; I.6.6 [Simulation and Modeling]: Simulation Output Analysis; I.3.8 [Computer Graphics]: Applications

1 INTRODUCTION

This work was motivated by the completion of a very high resolution global ocean simulation. The run covered 14 months of simulated ocean time, with 242 million grid points at each of 90 ocean depths, for a total of 22 billion grid points. Output was written at each hour of simulated time, for a total of 10311 time steps, each with 20 variables (5 3D fields and 15 2D fields) — constituting five petabytes of stored data. We first created movies showing the data, which we could do at full resolution on our display wall, the hyperwall [33]. However, we wanted to create a tool that allows browsing, side-by-side comparisons, and dynamic linked scatterplot brushing. In short, our aim was to allow the researchers to interact with their data in real time, not merely view it.

As a first step, the data were reformatted into multimedia files, each file corresponding to one display screen on the hyperwall, at one depth. The 32-bit floating point data were converted into 16-bit integers, and compressed using the FFV1 grayscale lossless multimedia codec [30]. Together, these operations reduced the data size by a factor of four. The operations also formatted the data so that playback only needs sequential file reads. This increases I/O performance when reading the files from disk since it minimizes seeks and allows prefetching.

A display program runs on each hyperwall node, and is capable of decoding and displaying two of these multimedia files simultaneously. We are typically able to stream at 24 frames per second, which is one simulated day per second. While the global view shown in Figure 2 is popular, in fact any node is capable of decoding any pair of files. This allows for the simple side by side comparison of any two quantities. However, since the scalar data are rendered on the fly, we are able to combine and compare the two data streams on the same screen in interesting ways: split screen, checkerboarding, blending, or subtracting. The color map for each movie can be changed interactively allowing dynamically chosen ranges to be highlighted. Histogram equalization can be done on a per-screen basis to bring out the greatest possible detail.

Additionally, instead of displaying the two multimedia files as portions of a global map, we can treat them simply as data values, and display them as a scatterplot. When animating, the scatterplots update with each new frame. Standard brushing techniques can be used to either select a region on a display, and observe the associated data values in the scatterplot, or select a set of data values on the scatterplot, and observe the points overlaid on an associated map.

These capabilities provide a rich and powerful tool set for the scientists to quickly home in on interesting features of the data, and to look for previously unseen correlations between various quantities, depths, and areas of the globe.

The remainder of the paper is structured as follows. We first describe some related work, and then describe the simulation that motivated this work. The following sections describe the application in detail, and then explain how it was implemented. The paper concludes with a short evaluation section, conclusions, and future work.

2 RELATED WORK

There is a significant body of work in related subjects: tiled displays; ocean visualization, and the similar atmospheric visualization; image compression; scatterplots; and high-dimensional visualization. Chupa *et al*. [15] describe a system that has several similarities to this paper: it uses compressed images on a tiled display to perform visualization of high-resolution ocean simulation data. This paper extends that earlier work with interactive colormaps, comparison features, and scatterplots. Another system using a tiled display for atmospheric visualization was described by Zudilova-Sienstra *et al*. [39], and included a user study. Tiled displays have also been used for visualizing high-resolution image collections [38].

Related work in the general area of visualization for ocean simulation output includes the paper by Butkiewicz and Ware [13], which describes a flow visualization system that runs on a tablet. The system by Köthur *et al*. [24] uses clustering to compare the output of different ocean models. Samsel *et al*. explore interactive colormapping [32] to see multiple data ranges in ocean salinity; our system also uses interactive colormapping so the entire range of salinity and other variables can be seen. Other ocean simulation visualization was described by Shen [34], who wrote about a system for visualizing a regional ocean model, and by Woodring [37], who focused

*e-mail: David.Ellsworth@nasa.gov
†e-mail:Chris.Henze@nasa.gov
‡e-mail:Bron.C.Nelson@nasa.gov

Figure 1: The hyperwall showing the application running, with the console at the left. The screens show horizontal velocity for one column of the globe; the columns left to right show successively lower depths. The orange regions are in the areas where the ocean is shallower than the depth shown in that column, and show the ocean bottom.

on the in situ analysis of ocean eddies. The paper by Brislawn *et al.* [12] explores compression techniques for ocean data with an emphasis on progressive transmission, which differs from this work because we use compression to save space and reduce bandwidth needs. Berres *et al.* [10] also investigated compression techniques for ocean visualization, with a focus on using video compression algorithms to compress a database of visualization images generated in-situ.

Visualization systems for atmospheric simulation are pertinent since both atmospheric and ocean simulation use multiple variables, include vector flow data, and have a 3D grid where the vertical axis is generally lower resolution than the others. Papers focusing on comparing the simulation models have been written by Biswas *et al.* [11] and Poco *et al.* [31]. Song *et al.* [35] describe a general purpose atmospheric visualization system, and present results for several visualization tasks. Techniques for combining visualization methods with multivariate atmospheric data have been proposed by Wong [36]. Finally, Dorier *et al.* [17] look at atmospheric simulation visualization from the in situ perspective with an emphasis on limiting the visualization cost.

The combination of scatterplots and brushing of multiple variables in an array of plots is time honored; some of the earlier descriptions were written by Becker and Cleveland [9] and Monmonier [29]. The technique was applied to cardiac simulations and linked with other visualization types by Gresh *et al.* [20]. Henze explored the use of linked scatterplots with derived variables for feature detection [21]. Kehrer *et al.* examined the use of scatterplots with statistical moments such as mean and variance [22]. Other works with scatterplots using transformed variables include those by Chan *et al.* [14] and Kumatani *et al.* [25].

The final category of related work is high-dimensional visualization. Most of the work previously cited are examples of this category, as ocean and atmospheric simulation data have multiple variables, plus the scatterplot works cited use high-dimensional data. Further information is contained in the survey by Kehrer and Hauser [23].

3 MITgcm Simulation

The driving problems and input data for this work come from high-resolution runs of the Massachusetts Institute of Technology general circulation model, or MITgcm [7, 27], as part of the ECCO2 project [1]. MITgcm is a finite volume numerical code for studying atmospheric or oceanic circulation on a rotating planet. The current work uses a configuration that produces state estimates of the Earth's global ocean circulation by solving the hydrostatic Boussinesq fluid flow equations on a realistic bathymetric domain, with surface forcing of momentum, mass, radiative and buoyancy fluxes taken from ERA-Interim re-analysis fields [16]. Model output is provided on staggered Arakawa C grids [8], configured as a 1/48-degree "Lat-Lon-Cap" global grid system [19], which has 12 4320 x 4320 x 90 grid point "facets" making up the "Lat-Lon" sectors, and a single 4320 x 4320 x 90 polar "Cap" covering the Arctic. The fixed-z^* (modified height coordinate [6]) model grid has 90 depth levels, and 242 million grid cells per horizontal level, for a total of 22 billion (wet and dry) grid cells.

The 1/48-degree model was initialized from a spun up 1/24-degree solution corresponding to September 10, 2011, and run on 30 to 70 thousand cores of the Pleiades supercomputer at NASA Ames for approximately 14 months of ocean time, ending on ocean date October 9, 2012. During these runs we emitted several 2D and 3D fields every ocean hour, for a total of 10311 time steps written to disk, constituting over 5 PB floating point data. The hourly outputs included all prognostic variables (horizontal velocity, sea surface height, temperature, salinity, and sea ice parameters), multiple diagnostic variables (vertical velocity, mixed layer thickness, bottom pressure, etc.), and several forcing fields sampled onto the model grid (wind stress, radiative fluxes, fresh water and salt fluxes, etc.). The hourly outputs were written to disk in 32-bit precision, and less frequent checkpoints in 64-bit precision, all with custom asynchronous-I/O code we designed originally for concurrent visualization [18]. The asynchronous-I/O code runs on 5 to 10% of the nodes, which are dedicated to I/O. Once the I/O nodes receive data from the compute nodes, they overlap domain recomposition and file writing (without an inline rendering step, in this case) almost entirely with the ongoing model computation.

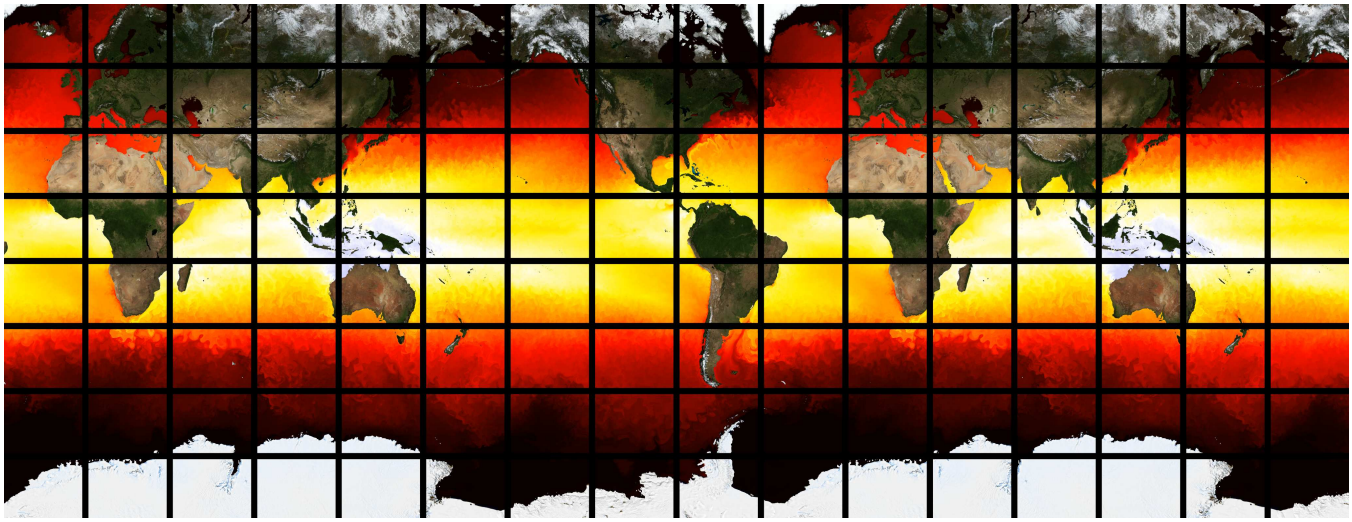


Figure 2: The hyperwall showing a global view of temperature using a heated-object color map.

The ocean scientists originally analyzed the data set by extracting small portions of the domain and transferring the extracted data to their location. They then used conventional visualization and analysis techniques to look at those data. These techniques made it impossible to get a high-resolution overview of the data set, and took much longer than doing the analysis with the new application.

4 Application Features

We currently convert 13 of the 15 2D fields and 23 of the 90 depths for each of the 5 3D fields into multimedia files, breaking the domain at a given depth into screen-sized tiles. Since the aspect ratios of the simulation domain and the hyperwall are different (the hyperwall is wider), we replicate part of the domain to make the aspect ratios match, and also remove the bottom 11% as it falls on the Antarctic continent. While replicating the domain increases the overall tile storage, the replication adds the important advantage that all of the ocean basins are contiguous. After removing the bottom, the remaining images are taller than the hyperwall, so we scale the images by 93%. After scaling the images, we cut them into 128 1728 x 1328 tiles. These tiles are larger than the 1600 x 1200 screen resolution because we include the pixels that are hidden by the display bezels. An option allows either displaying the center 1600 x 1200 of the images, to get a pleasing global view, or scaling the tiles down to fit the screen so all the data are visible. We don't currently support showing the Arctic area, as it is separate from the main domain, but doing so would be straightforward.

Displays showing a map view cover the land areas with satellite imagery, and the bare ocean floor areas with bathymetry. This imagery is not stored in the encoded multimedia files, but in separate overlay image files, one for each geographic location. Each file contains both the satellite and bathymetry imagery. A bit mask image for each geographic location and depth indicates whether the colormapped data or the overlay image should be displayed.

4.1 Layout Controls

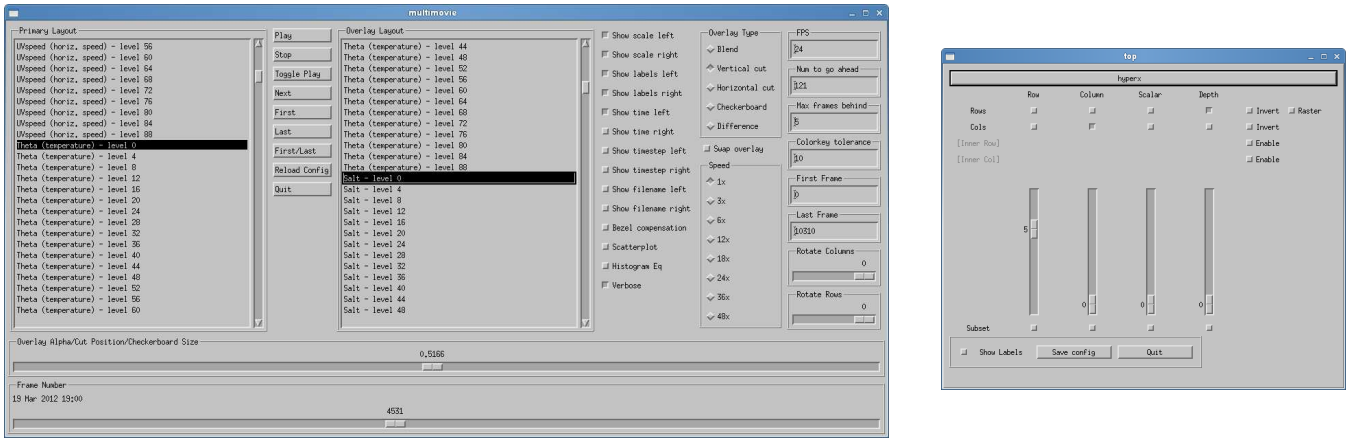
We have found that different analysis tasks need different layouts of tiles on the screen. Layouts are selected in large list selection boxes as shown in Figure 3(a). Most of the layouts are created with a short script, while others are created using an editor to create an input file to a utility. Creating a new layout is a quick process as only a single input file needs to be created; the multimedia files do not need to be changed. Every entry in a layout file contains the file name for the tile to be shown as well as whether that tile should be shown in a map or scatterplot view. We use file names to specify the tile because it gives complete flexibility in creating layouts. The layout creation script currently creates hundreds of layouts, which can be categorized as follows:

- A global view, with a layout for each scalar and depth (Figure 2).
- A view showing depths horizontally, where one column from the global view is replicated horizontally but with a different depth in each column; created for each scalar and column (Figure 1).
- A view showing depths vertically, similar to the above but with one row replicated instead one column replicated (Figure 4).
- A view with everything for one geographic location: all thirteen 2D scalars and all five 3D scalars at all 23 depths.
- An easier to use view of the above: all thirteen 2D scalars and all five 3D scalars at 16 depths, with each 3D scalar filling one row of displays (Figure 5).
- A view of the east or west half of the global view, with columns alternating between map and scatterplot views; a layout is produced for each scalar and depth.

Two additional layout controls are sliders: one that rotates the layout up and down, and a second that rotates the layout left and right. This makes it easier to see the top row of the layout, or to see the far right of the layout from the console located near the left edge of the screen array.

4.2 Interactive Layout Specification

A separate interface can be used to interactively specify layouts beyond the preconfigured list of layouts. The interface has been previously published as an interface for exploring families of parameterized simulations [33]. The interface treats the set of tiles as a 4D hypercube, with the four dimensions being the row (of the tile within the global view), column, scalar, and depth. The interface lets the user specify which two dimensions should be mapped to the rows and columns of the screens. The value to be used for two remaining dimensions are specified with a slider. This can be thought of as slicing the hypercube with a plane, with the elements of the



(a) (b)

Figure 3: User interface. (a) The main user interface window; the configuration shown is the one used to produce Figure 6. (b) The main window of the interactive layout interface; the settings shown were used to produce Figure 4.

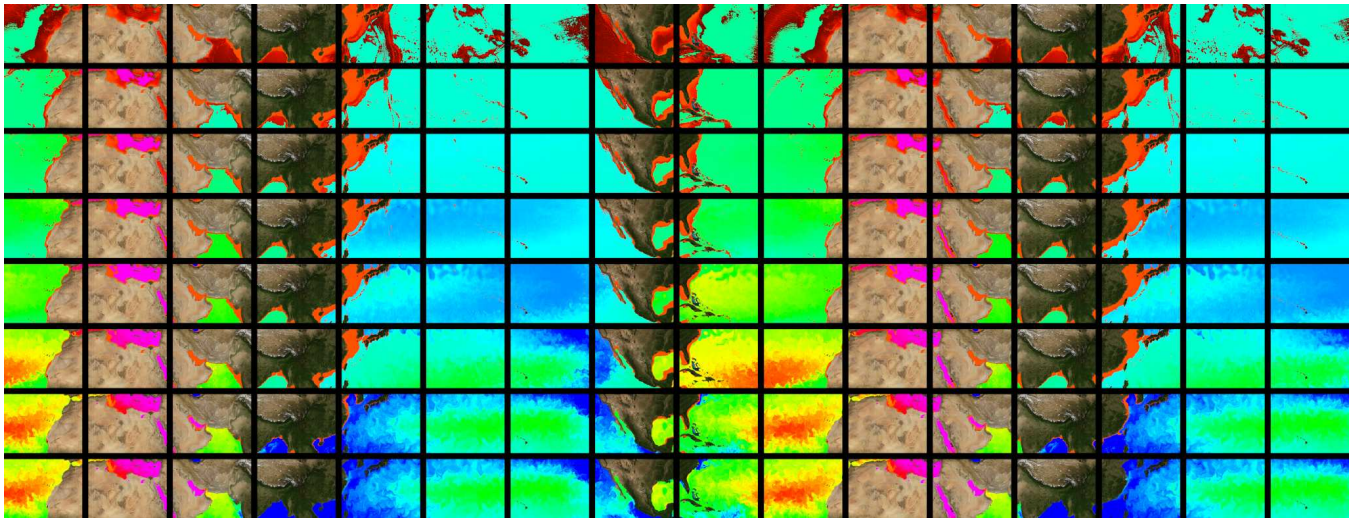


Figure 4: The hyperwall showing the layout created in Figure 3(b). The scalar shown is salt concentration; the surface is shown on the bottom row, and each higher row goes down three depth levels.

hypercube cut by the plane shown on the screens. The sliders control the position of the plane within the hypercube. See Figure 3(b) for an example.

A data dimension that is mapped to a display wall row or column can be modified in two ways. First, some of the items in that dimension can be omitted by calling up a dialog box and only selecting the wanted items. For example, if one of the mapped dimensions was 'scalar values', the tiles for salt and temperature could be omitted by not checking them in the dialog box list. Second, a stride or offset can be specified by using the second part of the interactive layout interface (not shown). Figure 4 shows the depth variable mapped to rows with a stride of 3.

4.3 Time Controls

The application has a simple set of time controls. There are the standard play and stop buttons, plus a slider to give quick access to any frame. The slider can respond to key presses, which will move backwards or forwards a frame at a time. The animation can be slowed down by entering a frames per second number in a text box, or the animation can be sped up by having it skip frames, which is controlled with a set of radio boxes. We have found that both speeding up and slowing down the animation is important as it makes it easier to pick out slow-moving or fast-moving features, respectively. Finally, if there is a time range of interest then text boxes can be used to set a first and last frame, and the application will then play only those frames.

4.4 Interactive Color Map Editing

Using actual (albeit reduced precision) data values in the multimedia files instead of RGB colors allows the color mapping to be changed interactively; the 16-bit values give sufficient precision for a large amount of contrast enhancement. Our color editor allows the colors to be edited, as well as the minimum and maximum scalar values of the color map. The editor allows opacity to be specified along with each color, plus contours can be added. The application maintains one color map for each scalar, which is initialized at start up from a configuration file. A selection of preconfigured color maps, such as grayscale, rainbow, heated object, etc., are available.

When editing a color map, the user selects the map to be edited using a menu. Changes made in a color map are immediately broadcast

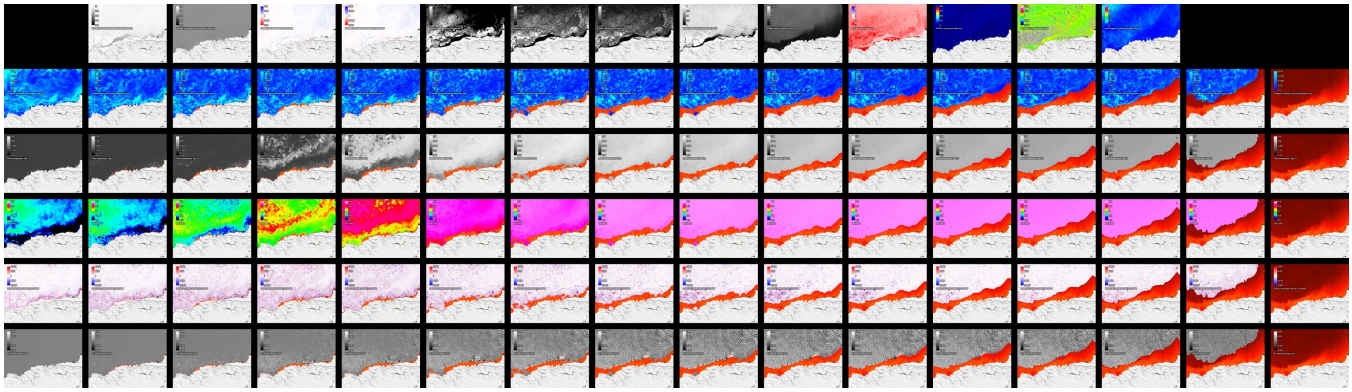


Figure 5: The hyperwall showing most of the tiles for geographic location (the top and bottom rows of the hyperwall are black and have been omitted). The tile shows the coast of Antarctica from approximately 125° to 155° West. The top row shows the 2D fields, while the remaining rows show the 3D fields: horizontal speed, temperature, salinity, horizontal vorticity, and vertical velocity. Only 16 of the 23 depths are shown for the 3D fields, with the depth increasing left to right. The tile shown is off the coast of Antarctica and shows circular plumes of cold brine falling due to freezing at wide cracks in the sea ice (they show up faintly on the screens showing temperature and salinity in the middle of the rows).

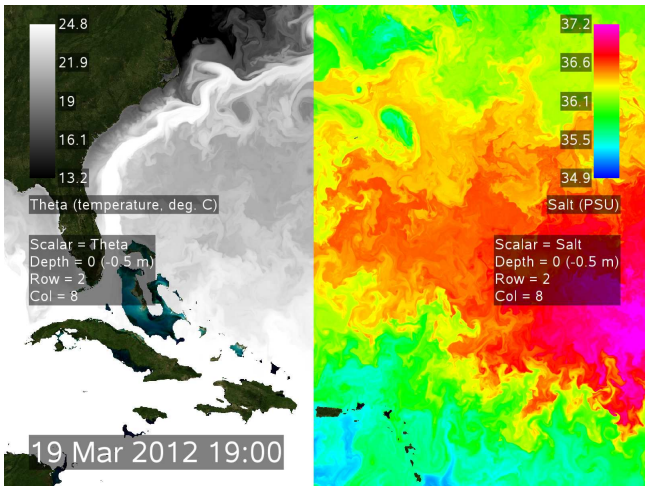


Figure 6: Example of a side-by-side comparison showing theta (temperature) and salt concentration at the ocean surface.

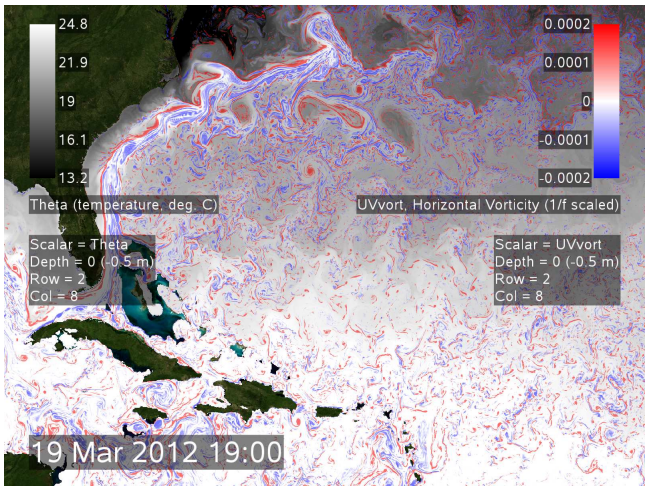


Figure 7: Example of an overlay comparison where the UV vorticity (the curl of the horizontal velocity) is overlaid over theta (temperature). In the regions where the UV vorticity values are low the temperature values show through completely. The example shows the correlation between regions of high temperature gradient and high vorticity.

to all the nodes and the affected displays are updated. The minimum and maximum color map values can be changed interactively by dragging a line in the interface, which makes it fast to find values that show the desired features.

The color maps can be optionally modified with histogram equalization to increase the contrast of the displayed images and allow fine details to be seen. The histogram equalization is done on a per-screen and per-time-step basis. This ensures that every screen shows the fine details, but it can distort the global view since the color map is no longer consistent across the screens. When a color scale is shown, we modify it to reflect the effect of the equalization step, so the color scale shows the correct local color mapping.

4.5 Single-Screen Comparison

In addition to comparisons on adjacent screens, comparisons can be on a single screen: the application can put two different tiles on a single screen. The different tiles are chosen by selecting a layout from the two large selection lists in Figure 3(a). The two tiles can be displayed four different ways. They can be displayed by splitting the screen, either horizontally or vertically, with the location of the split controlled by a slider (see Figure 6). Or, they can be displayed in a checkerboard pattern; the slider here controls the size of the squares. The two fields can be subtracted, and the result colored with a color map; this technique is especially useful when two different depths are subtracted. Finally, the two tiles can be blended, with the alpha of one controlled using the color map. In this case, the slider controls a separate alpha value that is multiplied with the map alpha value to get the final value. We have found that using blending with alpha works well when one tile is overlaid with the high values of the other tile; see Figure 7 for an example.

4.6 Annotations

Each screen can be annotated; the annotations are selectively enabled to reduce screen clutter. First, each of the two color maps used on the screen can be shown. Showing the color map is important to allow quantitative analysis. Second, the image row, column, scalar value, and depth can optionally be shown. We have found this annotation to be important as the flexibility in specifying tile layouts makes it easy to get confused about what a given screen is showing. Third,

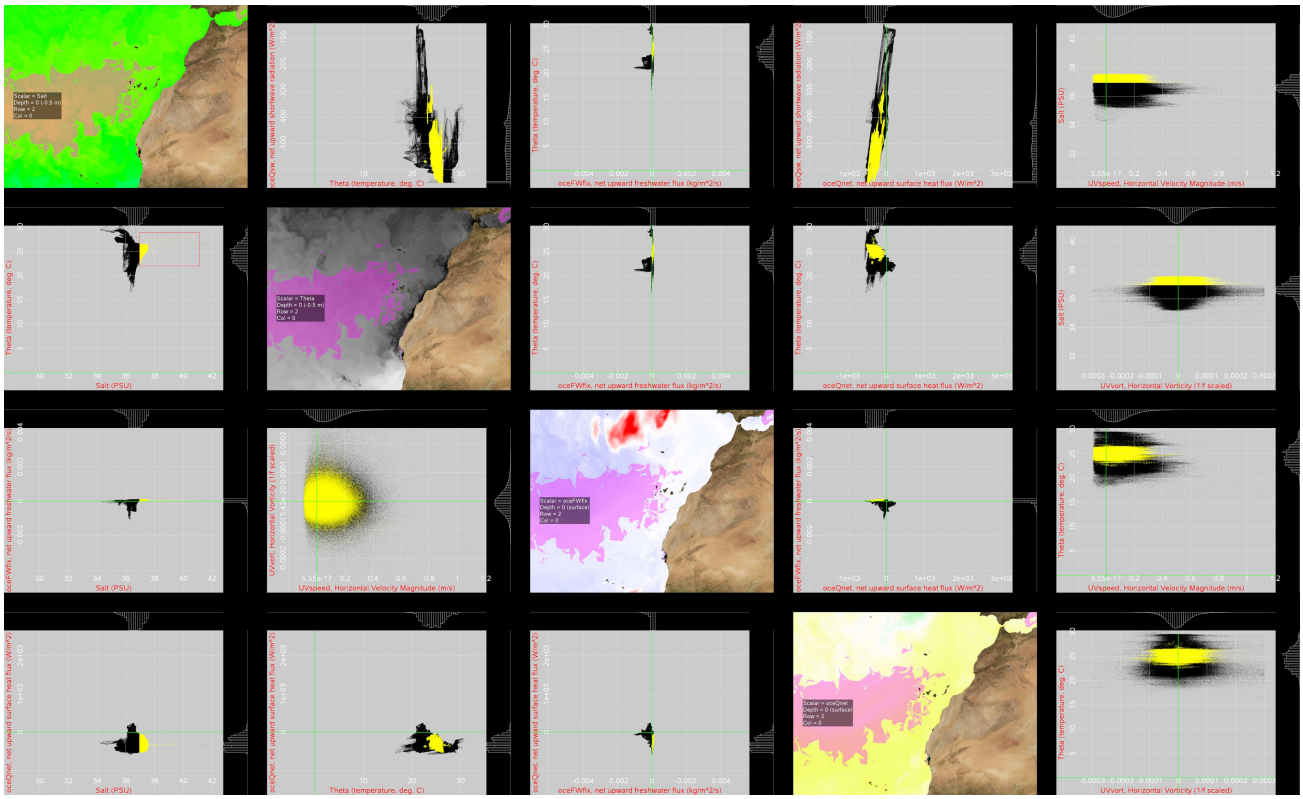


Figure 8: A portion of a custom layout showing a 4 by 5 array of scatterplots with some map view displays. All of the displays show data for the same tile (geographical location), the eastern Atlantic ocean off North Africa. The selection box is on the screen in the second row, first column.

the current time can be displayed. Examples of these annotations are shown in Figures 6 and 7. While the annotations are currently sized so they can be seen from a distance, we plan to add an option to decrease the annotation size to reduce screen clutter.

4.7 Scatterplots

Instead of showing the 16-bit scalars in map view, the application can display them as scatterplots (see Figure 8). The scatterplot view shows the point data along with a histogram for each scalar shown at the edge of the screen.

With scatterplots, the user interacts with the hyperwall displays instead of interacting with a user interface on the console. A separate program uses the X11 XTest library to allow the mouse cursor to move off the console and across the screens of the hyperwall, and to forward key presses and mouse events to the remote display.

Selecting a scatterplot view is done either via the layout file, or by a key press sent to a remote display. The scatterplot view can be customized by zooming and panning using the mouse, which allows a small region to be examined in detail.

All screens (ones in both the map and scatterplot view) respond to rectangular selections made via the mouse. A selection is shared between all the nodes displaying tiles from the same combination of image row, image column, and depth. We call the set of nodes a *node group*, in which we implement the well-known linked brushing operation [9]. A node group can only have one selection; making a new selection clears the old one. However, each different node group can have a different selection. The linked brushing is implemented by calculating a bitmap that indicates whether each point is in the current selection, and then sharing the bitmap with all of the nodes in the node group. Distributing the selection bitmap occurs fast enough that there is no noticeable lag, although when animating the selections may arrive one frame behind. Selections on the scatterplot displays can either be evaluated at the time of the selection, with the selected points held constant until another selection is made, or they can be evaluated with every new time step.

5 Implementation

The application runs on our hyperwall, a display wall with 128 screens in 8 rows and 16 columns (see Figure 1). Each screen is connected to a separate workstation-class system that has two 10-core 2.8 GHz E5-2680v2 Intel Xeon processors, 64 GB of memory, and a GeForce GTX 780 Ti GPU. The nodes are interconnected with a FDR Infiniband network, and use the network to communicate with a 30 GB/second (peak) Lustre file system. Each node also has a 2 TB PCIe NVMe SSD. All the SSDs are aggregated together with Excelero software to form a fast, low latency 250 TB file system. Finally, a separate but similarly-configured system is used as a console.

5.1 Architecture

The application runs on the console and hyperwall nodes. A script is used to start all the processes and propagate the initial configuration. The main process on the console runs the user interface, sends and receives the messages used to synchronize screen updates, and runs a (non-X11) event loop that determines the overall action of the application. The process on each node has threads that decode the multimedia file, draw new frames, and handle the scatterplot bitmask communication.

When animating, the screen updates are synchronized using a combination of a barrier and synchronized clocks. The decision to draw a new frame starts when the all nodes have finished decoding the new frame's image. Each node sends a "ready" message to the console, and when all of the messages have arrived, the console

determines and broadcasts a time in the near future when the nodes should start displaying a new frame. That time is the later of 1) the time for the next frame, assuming the application is in play mode, and 2) a fixed time in the future (5 ms), which gives sufficient time for the message to propagate and be handled. When a node receives the message telling it the time for the new frame to be displayed, it draws the frame into the back frame buffer, waits until the specified time (if it has not already past), and then displays the frame (swaps the front and back frame buffers).

The synchronization description in the previous paragraph implies that the next frame is displayed only when all the nodes are ready, but this is not necessarily true. Instead, we use a flexible barrier, where nearly all the nodes (a configurable number) need to be ready before proceeding, but with the proviso that a node can only be a limited number of frames behind the others. This flexible barrier minimizes the stuttering that occurs when the disk reads do not complete in time, but does mean that some displays do not display the current frame. We find that the flexible barrier improves the overall user experience.

On the console, one thread in the process runs the user interface, which is implemented using X11 and Motif. A second thread handles the main application event loop, which handles events generated by the user interface plus an event generated when all the nodes have indicated that they are ready to draw the next frame. We use an event loop to simplify the control code as it allows the asynchronous incoming events to be serialized.

The console also has separate processes and windows for the interactive layout interface. Layouts from each process are sent to the main process over a pipe. We generally have two instances of the interface running since two tiles per display can be decoded.

The hyperwall node portion of the application decodes the frames and updates the display. The decoding is done up to 5 frames in advance of the display update. Working ahead of the update gives a buffer to isolate frame update from delays in reading data from disk. The console portion of the application sends separate command streams to the decoders and the display code to facilitate the advance decoding. The advance decoding means that the display code must usually discard decoded frames when performing seeking or changing to a new tile. We use an OpenGL GLSL fragment shader to draw frames: it performs the colormap look up, combines the two images, does the histogram equalization when requested, and overlays the land/bathymetry image.

The application uses ZeroMQ for the communication to share scatterplot selections [3]. Scatterplot selections are sent to the console using Push-Pull ZeroMQ sockets. Selections are sent the appropriate nodes (the ones with the correct image row, column, and depth) using Publish-Subscribe sockets. Those sockets allow filters to be established on the subscribe (receiving) end so that only the desired messages are sent. Messages to clear the current selection when a new one is made are also sent via the same ZeroMQ sockets.

5.2 Encoding

We use the FFV1 codec to compress the 16-bit input images [30]. We chose it because it is one of very few that can losslessly encode 16-bit data, and because it is available from the open source FFMPEG project [5]. While we could store the raw FFV1 bitstream in a file, we chose to use a NUT container [4] because it has a frame index that allows for efficient seeks within the multimedia file, and it is also supported by the FFMPEG libraries. We had to modify the NUT code slightly for it to always emit a syncpoint startcode before each frame, as some low-complexity image sequences that compressed well did not have all of those startcodes, which are needed for efficient seeking.

We did some investigation into compression techniques; the results are in Table 1. The table has separate values for wet cells (cells containing water) because one could compress the data by only saving the wet cell data, and consulting an index to decompress those data to a regular grid at run time. The "wet cells" values give compression ratios assuming that was done.

Table 1: Compression statistics. The "All cells" line gives the compressed file sizes for several compression options as a percentage of the original 16-bit data sizes. The "Wet cells" line gives the percentages relative to the size of the 16-bit data without counting data for the dry cells (cells that contain land).

		FFV1 Codec			
Reference	PNG	16-bit	14-bit	12-bit	10-bit
All cells	36%	31%	23%	17%	11%
Wet cells	66%	57%	43%	31%	21%

Table 2: Per-field compression statistics given as a percentage of the 16-bit data sizes. See above for the definition of "Wet Cells" and "All Cells". Fields listed in bold are 3D.

Field	Wet Cells	All Cells	Field	Wet Cells	All Cells
Eta	26.5%	17.0%	**Salt**	31.0%	16.2%
KPPhbl	47.3%	30.3%	SIarea	7.88%	5.05%
oceFWflx	22.3%	14.3%	SIheff	8.82%	5.65%
oceQnet	32.8%	21.0%	SIhsalt	8.64%	5.54%
oceQsw	22.4%	14.4%	SIhsnow	7.50%	4.81%
oceSflux	6.05%	3.88%	**Theta**	39.6%	20.8%
oceTAUspeed	26.3%	16.9%	**UVspeed**	69.2%	36.3%
oceTAUvort	51.2%	32.8%	**UVvort**	82.2%	43.1%
PhiBot	51.5%	33.0%	**W**	85.8%	45.0%
Overall	57.0%	30.6%			

Overall, compressing 16-bit data resulted in cutting the size by a factor of 3 (31% of the original size), or reducing the size to 57% of the original size of the wet cells. Reducing the data precision by scaling the data so that some number of the top bits were always zero reduced the data size even further. This could be helpful if a further reduction of size or bandwidth was necessary. We also tried using the PNG compression algorithm instead of the FFV1 codec, but the file sizes were larger.

The compression ratios varied significantly by field, as detailed in Table 2. The W (vertical velocity) field was very complex, which meant that the file sizes were 45% of the original file size, or 86% of the size of the wet cells. On the other hand, the sea ice fields (the ones that start with SI) compressed very well since most of the ocean is free of ice. Those fields compressed to 5 to 6% of the original file size, or 8 to 9% of the wet cell data size. Overall, the compressed 16-bit data occupies 237 TB.

Extracting the values from the original data files and encoding them into multimedia files took several thousand node hours. The run time was dominated by the time required to read the original data from disk (over 1 PB).

Our research did not focus on compression. Other techniques are likely to give higher compression rates with errors rates comparable to the ones introduced by the 16-bit quantization. One possibility is the lossy floating point array compression proposed by Lindstrom [26]. However, the errors introduced by 16-bit quantization have the desirable property that the errors are apparent, for example, when a scatterplot is highly enlarged to show a very small detail. Compression techniques that make use of the data's temporal coherence are especially promising, but would complicate seeking within the multimedia files.

6 Evaluation

We measured the performance of the application both when reading from SSDs and when reading from our Lustre file system that has

Figure 9: Photo from the first visit of domain scientists.

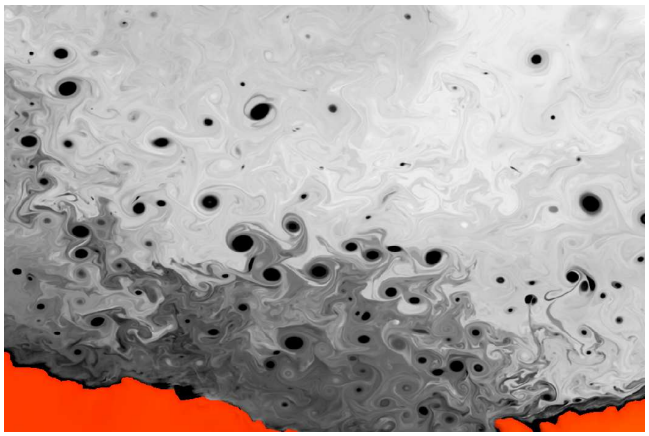


Figure 10: Compact eddies off the coast of Antarctica at 850 m depth produced from sinking cold brine plumes. The orange is the ocean bottom.

spinning disks. The file systems were otherwise quiet during our measurements. When the system is reading data off of SSDs it is able to reliably show successive time steps at our desired 24 time steps per second. We are able to maintain that frame rate when the layouts show tiles that have the lowest compression, which requires 15 GB/s.

The 24 Hz frame rate is also maintained when the system is reading data off of our Lustre file system's disks, although not as smoothly as with the SSDs. We found that the performance with Lustre was sensitive to how the multimedia files were striped on the Lustre Object Store Targets (OSTs), which are RAID disk arrays at our facility. The striping needed to be done so that, for every layout, the bandwidth required from each OST was not higher than could be supported by it. We wrote a program that takes into account all of the layouts and file sizes and calculates a near-optimal striping configuration. The generated configuration requires bandwidth from the OSTs that is only 10% higher than the theoretical optimum value.

After observing domain scientists work with the application, we believe that its most important features are:

- A high resolution display which could show a complete global view at once.
- Side-by-side and same-screen comparisons that let correlations between fields to be found.
- Interactive controls allowing layouts to be quickly selected.
- Time controls that could speed up time to make slow-moving features be apparent, and slow down time to let fast-moving features be studied.
- Scatterplot brushing that enabled features to be identified.

A different and arguably more important way to evaluate the system is user feedback. We have had several full work days with domain scientists who uniformly found the application to be very useful. Several of the scientists have come back for repeat visits, which we feel is significant as all of our visitors have had to take the trouble of traveling by air in order to visit. Once scientist [28] told us that the application "has revolutionized the way that we are able to utilize the model output and carry out scientific investigations. It allows us to explore the solution for new (as opposed to predetermined) physical phenomena. The viewer simultaneously allows us to look at global phenomena as well as detailed processes."

The scientists have found several features in the data that had not been seen before. For example, one feature was cold brine plumes rejected from freezing water at wide cracks in the sea ice, which descended until reaching neutral buoyancy where they produce compact eddies due to cyclogeostrophic arrest (see Figure 10). Investigating these features revealed overshooting and oscillations around the neutrally buoyant level, as well as unexpected positive and negative heat fluxes on opposite sides of the associated sea ice cracks. The scientists were also able to investigate seasonality of the surface and deep vorticity fields, and their interaction and modification by internal waves, which is of much importance to the upcoming Surface Water and Ocean Topography mission [2]. Finally, the scientists have found issues with the simulation's bathymetry data as well as the temporal resolution of the wind data used to drive the ocean surface.

7 Conclusions and Future Work

We have presented the design of an application that effectively allows interactive visualization and exploration of a 5 PB ocean simulation data set. The application reformats the data so it can be presented on the 245 million pixel hyperwall at near-native resolution at 24 time steps per second. It has numerous features for browsing, comparison and quantitative analysis. Domain scientists have found the application to be a useful tool and have found several new features in the data.

We have a long list of features that have been requested to be added to the application. The features include an interactively-specified vertical cut. Vertical cuts would require that all the layers be decoded. Since we cannot decode all the layers for all the displays, we plan to implement this by restricting the cut to a single tile, and using multiple nodes to decode all of the layers for that tile, one tile per node. Another requested feature is temporal filtering, which would remove the diurnal cycle or tidal movements, and would let the remaining features be seen more readily.

Acknowledgments

The authors wish to thank the ECCO2 project Co-PIs, Dimitris Menemenlis and Chris Hill, for the use of their data and for their enthusiastic collaboration. We would also like to thank Timothy Sandstrom for developing his GLX application environment that includes the color editor that we use, as well as developing the interactive layout interface.

References

[1] ECCO2 project home page. `http://ecco2.org`.
[2] SWOT home page. `https://swot.jpl.nasa.gov`.

[3] ZeroMQ distributed messaging. http://zeromq.org.

[4] NUT open container format 20131223. http://www.ffmpeg.org/~michael/nut.txt, 2013.

[5] FFmpeg home page. http://ffmpeg.org/, 2017.

[6] A. Adcroft and J.-M. Campin. Rescaled height coordinates for accurate representation of free-surface flows in ocean circulation models. *Ocean Modelling*, 7(3):269–284, 2004.

[7] A. Adcroft, C. Hill, J.-M. Campin, J. Marshall, and P. Heimbach. Overview of the formulation and numerics of the MIT GCM. In *Proceedings of the ECMWF seminar series on Numerical Methods, Recent developments in numerical methods for atmosphere and ocean modelling*, pp. 139–149, Sept. 2004.

[8] A. Arakawa and V. R. Lamb. Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics: Advances in Research and Applications*, 17:173–265, 1977. doi: 10.1016/B978-0-12-460817-7.50009-4

[9] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987. doi: 10.2307/1269768

[10] A. S. Berres, T. L. Turton, M. Petersen, D. H. Rogers, and J. P. Ahrens. Video Compression for Ocean Simulation Image Databases. In K. Rink, A. Middel, D. Zeckzer, and R. Bujack, eds., *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2017. doi: 10.2312/envirvis.20171104

[11] A. Biswas, G. Lin, X. Liu, and H.-W. Shen. Visualization of time-varying weather ensembles across multiple resolutions. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):841–850, Jan. 2017. doi: 10.1109/TVCG.2016.2598869

[12] C. M. Brislawn, J. L. Woodring, S. M. Mniszewski, D. E. DeMarle, and J. P. Ahrens. Subband coding for large-scale scientific simulation data using JPEG 2000. In *2012 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pp. 201–204. IEEE, Apr. 2012. doi: 10.1109/SSIAI.2012.6202488

[13] T. Butkiewicz and C. Ware. Multi-touch 3D exploratory analysis of ocean flow models. In *OCEANS 2011 MTS/IEEE KONA*, pp. 1–10. IEEE, Sept. 2011. doi: 10.23919/OCEANS.2011.6107079

[14] Y.-H. Chan, C. D. Correa, and K.-L. Ma. Flow-based scatterplots for sensitivity analysis. In *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pp. 43–50. IEEE, Oct. 2010. doi: 10.1109/VAST.2010.5652460

[15] M. A. Chupa, R. J. Moorhead, J. A. Mogill, J. F. Shriver, D. W. Irby, and P. M. Flynn. EnVis/Hum: high-resolution ocean model visualization and display. In *Oceans 2003. Celebrating the Past... Teaming Toward the Future (IEEE Cat. No. 03CH37492)*, vol. 2, pp. 1030–1031. IEEE, Sept. 2003. doi: 10.1109/OCEANS.2003.178478

[16] D. Dee, S. Uppala, A. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. Balmaseda, G. Balsamo, P. Bauer, et al. The era-interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):553–597, 2011. doi: 10.1002/qj.828

[17] M. Dorier, R. Sisneros, L. B. Gomez, T. Peterka, L. Orf, L. Rahmani, G. Antoniu, and L. Bougé. Adaptive performance-constrained in situ visualization of atmospheric simulations. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 269–278. IEEE, Sept. 2016. doi: 10.1109/CLUSTER.2016.25

[18] D. Ellsworth, B. Green, C. Henze, P. Moran, and T. Sandstrom. Concurrent visualization in a production supercomputing environment. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):997–1004, 2006. doi: 10.1109/TVCG.2006.128

[19] G. Forget, J.-M. Campin, P. Heimbach, C. Hill, R. Ponte, and C. Wunsch. ECCO version 4: an integrated framework for non-linear inverse modeling and global ocean state estimation. *Geoscientific Model Development*, 8(10):3071–3104, 2015. doi: 10.5194/gmd-8-3071-2015

[20] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data. In *Proceedings Visualization 2000*, pp. 489–492. IEEE Computer Society Press, 2000. doi: 10.1109/VISUAL.2000.885739

[21] C. Henze. Feature detection in linked derived spaces. In *Proceedings Visualization'98*, pp. 87–94. IEEE Computer Society Press, Oct. 1998. doi: 10.1109/VISUAL.1998.745289

[22] J. Kehrer, P. Filzmoser, and H. Hauser. Brushing moments in interactive visual analysis. *Computer Graphics Forum*, 29(3):813–822, 2010. doi: 10.1111/j.1467-8659.2009.01697.x

[23] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE transactions on visualization and computer graphics*, 19(3):495–513, Mar. 2013. doi: 10.1109/TVCG.2012.110

[24] P. Köthur, M. Sips, H. Dobslaw, and D. Dransch. Visual analytics for comparison of ocean model output with reference data: Detecting and analyzing geophysical processes using clustering ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1893–1902, Dec. 2014. doi: 10.1109/TVCG.2014.2346751

[25] S. Kumatani, T. Itoh, Y. Motohashi, K. Umezu, and M. Takatsuka. Time-varying data visualization using clustered heatmap and dual scatterplots. In *2016 20th International Conference Information Visualisation (IV)*, pp. 63–68. IEEE, July 2016. doi: 10.1109/IV.2016.50

[26] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, Dec. 2014. doi: 10.1109/TVCG.2014.2346458

[27] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey. A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, 102(C3):5753–5766, 1997. doi: 10.1029/96JC02775

[28] D. Menemenlis. (ECCO2 Co-PI) personal communication, June 2017.

[29] M. Monmonier. Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. *Geographical analysis*, 21(1):81–84, 1989. doi: 10.1111/j.1538-4632.1989.tb00879.x

[30] M. Niedermayer. FFV1 video codec specification. http://www.ffmpeg.org/~michael/ffv1.html, 2013.

[31] J. Poco, A. Dasgupta, Y. Wei, W. Hargrove, C. Schwalm, R. Cook, E. Bertini, and C. Silva. SimilarityExplorer: A visual inter-comparison tool for multifaceted climate data. *Computer Graphics Forum*, 33(3):341–350, 2014. doi: 10.1111/cgf.12390

[32] F. Samsel, S. Klaassen, M. Petersen, T. L. Turton, G. Abram, D. H. Rogers, and J. Ahrens. Interactive colormapping: Enabling multiple data range and detailed views of ocean salinity. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pp. 700–709. ACM, New York, NY, USA, 2016. doi: 10.1145/2851581.2851587

[33] T. A. Sandstrom, C. Henze, and C. Levit. The hyperwall. In *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization - CMV 2003*, pp. 124–133, July 2003. doi: 10.1109/CMV.2003.1215010

[34] Y. Shen, J. A. Austin, J. R. Crouch, and M. S. Dinniman. Interactive visualization of regional ocean modeling system. In *Proceedings of the IASTED International Conference on Graphics and Visualization in Engineering*, pp. 74–82. Acta Press, 2007.

[35] Y. Song, J. Ye, N. Svakhine, S. Lasher-Trapp, M. Baldwin, and D. Ebert. An atmospheric visual analysis and exploration system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1157–1164, 2006. doi: 10.1109/TVCG.2006.117

[36] P. C. Wong, H. Foote, D. L. Kao, R. Leung, and J. Thomas. Multivariate visualization with data fusion. *Information Visualization*, 1(3-4):182–193, Dec. 2002. doi: 10.1057/PALGRAVE.IVS.9500024

[37] J. Woodring, M. Petersen, A. Schmeiβer, J. Patchett, J. Ahrens, and H. Hagen. In situ eddy analysis in a high-resolution ocean climate model. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):857–866, Jan. 2016. doi: 10.1109/TVCG.2015.2467411

[38] S. Yamaoka, K.-U. Doerr, and F. Kuester. Visualization of high-resolution image collections on large tiled display walls. *Future Generation Computer Systems*, 27(5):498–505, May 2011. doi: 10.1016/j.future.2010.12.005

[39] E. Zudilova-Seinstra, J. Adriaanse, A. Berg, Y. Xiao, and E. ten Thij. Visualizing climate data on a tiled panel display: A user-centered approach. In *IADIS International Conference on Computer Graphics and Visualization 2008*, pp. 197–202. IADIS, 2008. doi: 11245/1.285227